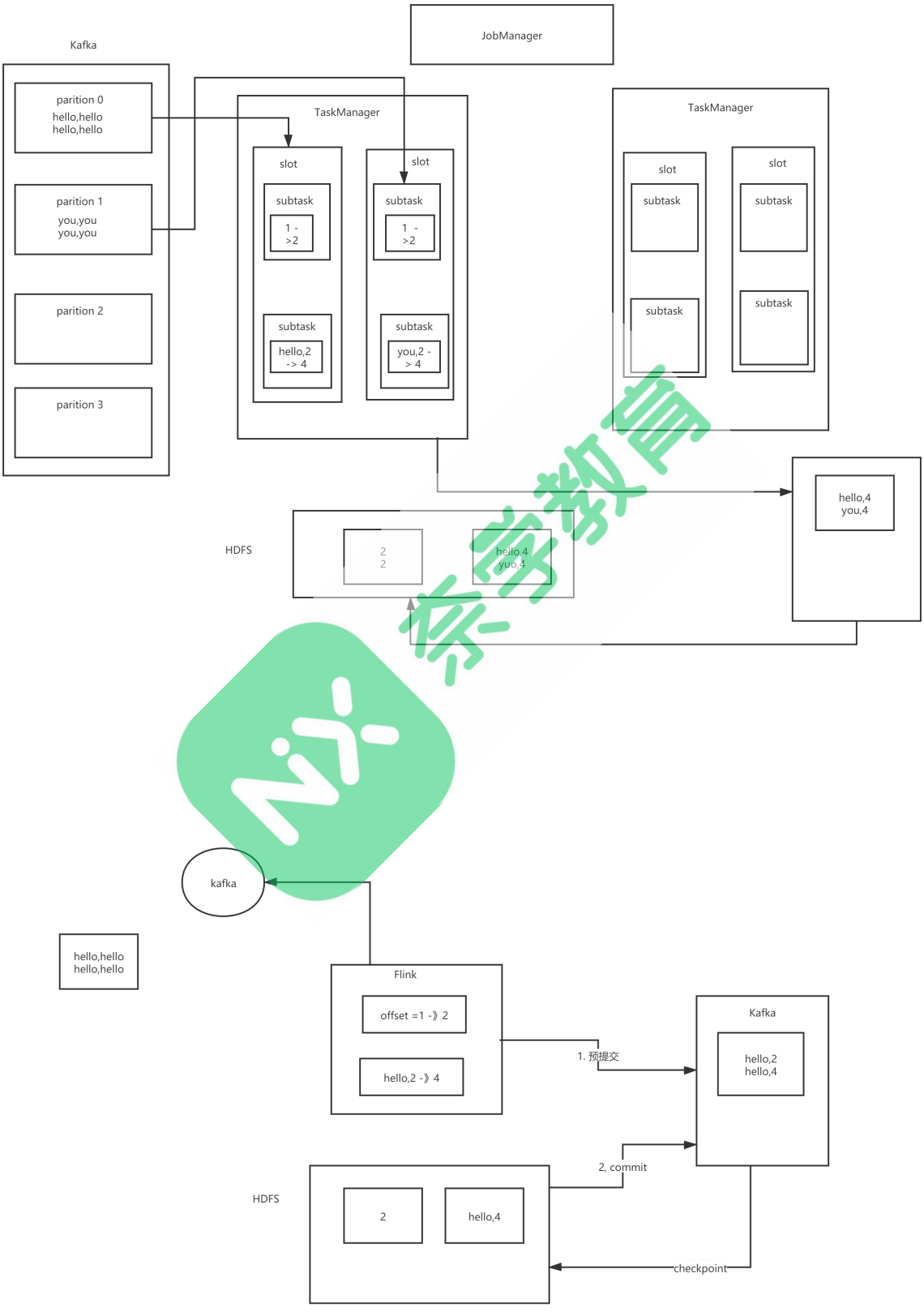
Flink面试题

1. Flink从Kafka读取数据，把处理结果写入Redis/HBase/Kafka，如何保证Exactly-once？



参考建议：1. 把Redis/HBase方案说清楚，通过atlest once + 幂等设计 = Exactly-once 2. Kafka的两阶段提交方案描述清楚。

2. 如何设计一个SQL on Stream的平台？ 概率比较小

用我们之前讲的SQL on Stream方案的内容回答就行，说思路就可以。

参考答案：这个问题，其实就是面试官，他想做类似的平台，自己不知道怎么做，想通过你启发一下他，需求一些经验。
就按我们之前讲的SQL on Stream平台的方案讲就行。

3. Flink相比SparkStreaming有什么区别

1. 架构模型

Spark Streaming 在运行时的主要角色包括：Master、Worker、Driver、Executor

Flink 在运行时主要包含：Jobmanager、Taskmanager、 Client 、Slot。

2.任务调度

Spark Streaming 连续不断的生成微小的数据批次，构建有向无环图 DAG，Spark Streaming 会依次创建 DStreamGraph、JobGenerator、JobScheduler。

Flink 根据用户提交的代码生成 StreamGraph，经过优化生成 JobGraph，然后提交给 JobManager 进行处理，JobManager 会根据 JobGraph 生成 ExecutionGraph，ExecutionGraph 是 Flink 调度最核心的数据结构，JobManager 根据 ExecutionGraph 对 Job 进行调度。

3.时间机制

Spark Streaming 支持的时间机制有限，只支持处理时间。

Flink 支持了流处理程序在时间上的三个定义：处理时间、事件时间、注入时间。同时也支持 watermark 机制来处理滞后数据。

4. 容错机制

对于 Spark Streaming 任务，我们可以设置 checkpoint，然后假如发生故障并重启，我们可以从上次 checkpoint 之处恢复，但是这个行为只能使得数据不丢失，可能会重复处理，不能做到恰一次处理语义，如果我们想要实现Exactly-once，需要自己实现。

Flink 内部提供了Exactly-once实现，使用起来较为方便

5. 还可以有其他角度

state，window等等

参考建议：这个是一个非常灵活的面试题，也是一个比较好的面试题，一方面可以考察出来你对Spark Streaming和Flink的了解的程度。一方面可以考察出来你对实时计算理解的程度。这个题的回答答案不唯一，但是回答的适合思路一定要清晰。千万别柔在一起回答。

4. Flink的三种语义是什么?

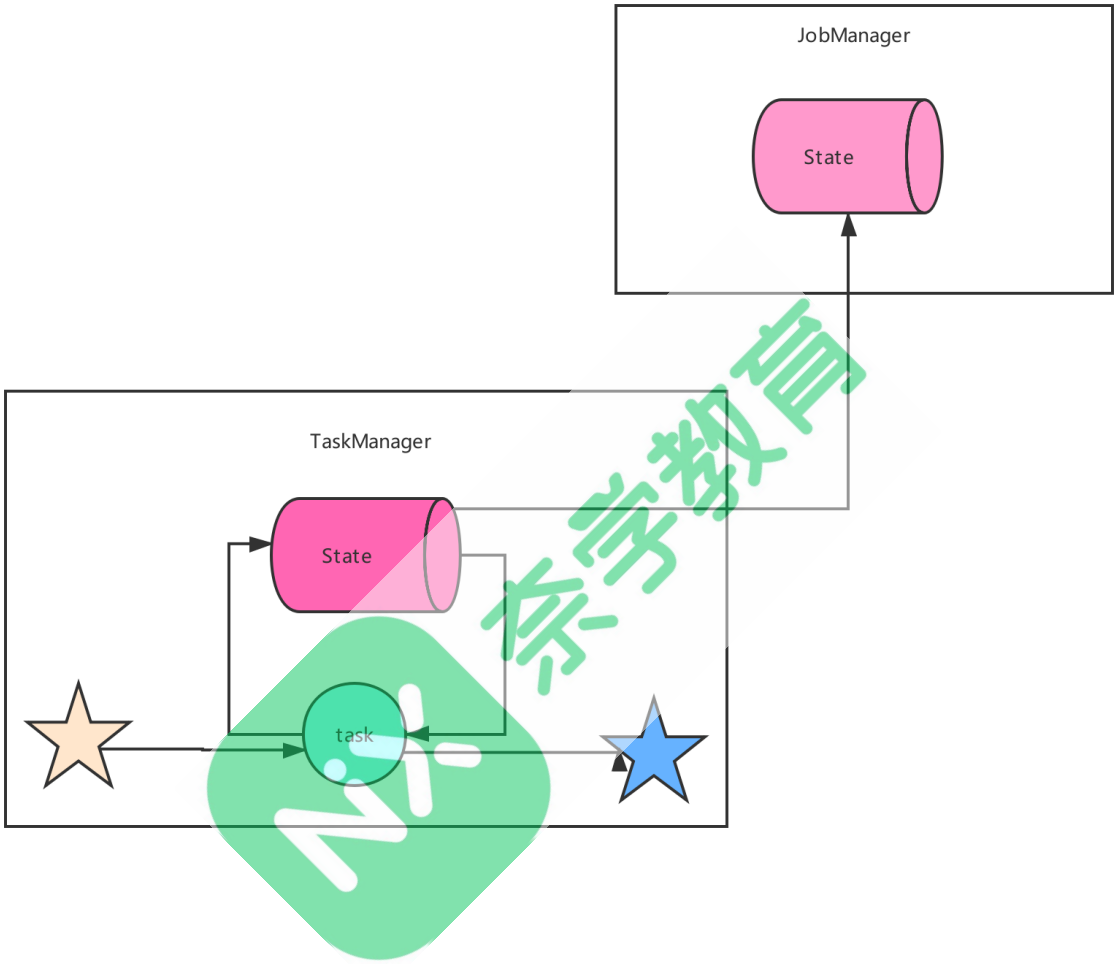
1. Event Time：这是实际应用最常见的时间语义。
2. Processing Time：没有事件时间的情况下，或者对实时性要求超高的情况下。
3. Ingestion Time：存在多个 Source Operator 的情况下，每个 Source Operator可以使用自己本地系统时钟指派 Ingestion Time。后续基于时间相关的各种操作，都会使用数据记录中的 Ingestion Time。

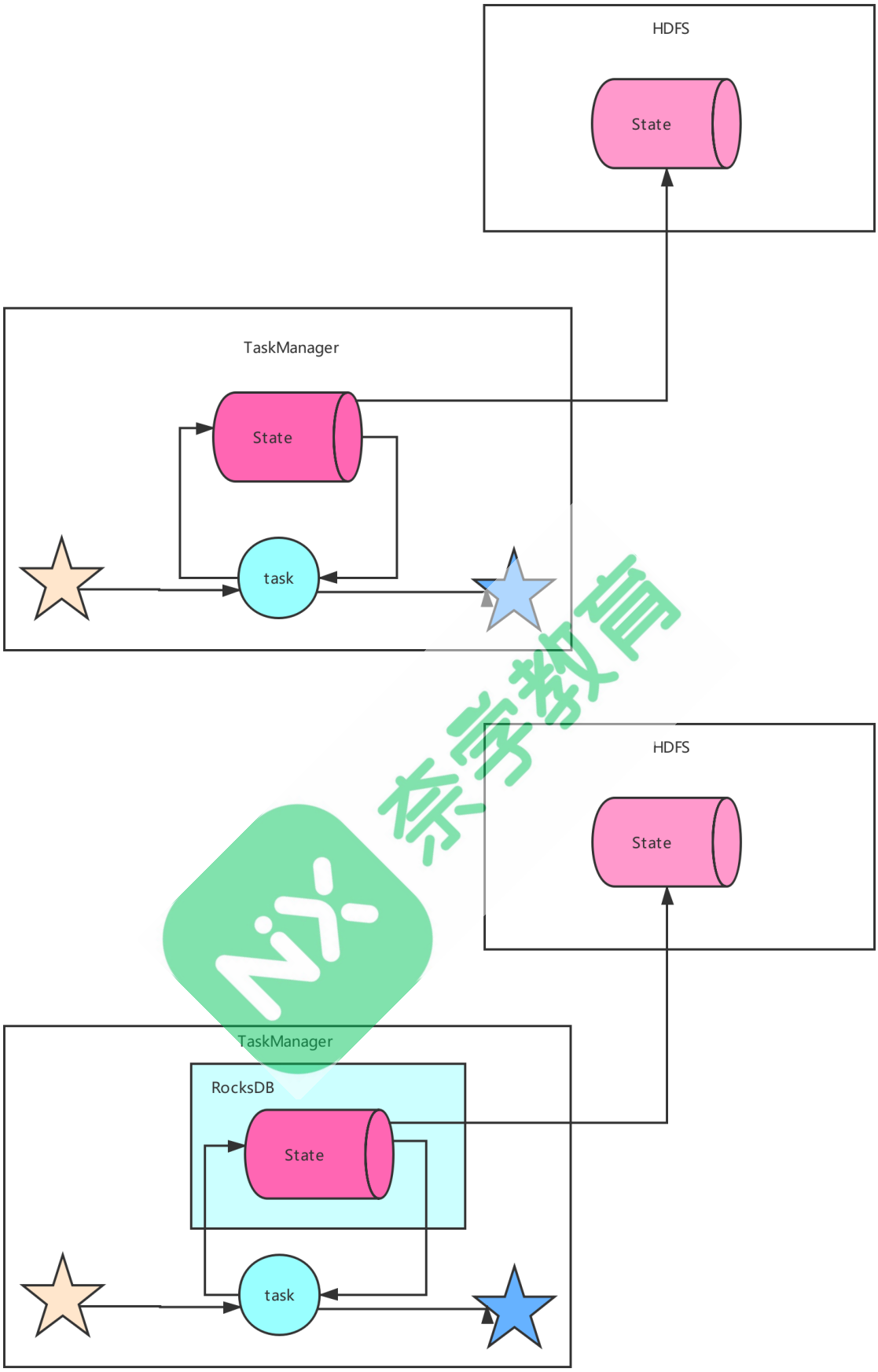
5. Flink场景的重启策略有哪几种?

Flink 实现了多种重启策略，常见的有如下三种：
固定延迟重启策略（Fixed Delay Restart Strategy）
故障率重启策略（Failure Rate Restart Strategy）
没有重启策略（No Restart Strategy）

6. 说说Flink的状态存储

Flink提供了三种状态存储方式：MemoryStateBackend、FsStateBackend、RocksDBStateBackend。





参考建议：不要只是简单的说MemoryStateBackend、FsStateBackend、RocksDBStateBackend这样的三个名词，还要说一下这三个backend的特点和适用场景，才是完整的答案。

7.Flink 中 exactly-once 语义是如何实现的，状态是如何存储的?

1）Flink内部 依靠 checkpoint 机制来实现 exactly-once 语义，
2）如果要实现端到端的 exactly-once，还需要外部 source 和 sink 满足一定的条件。状态的存储通过状态后端来管理，Flink 中可以配置不同的状态后端。

参考建议：回答的时候从两个角度回答，一个是flink内部是如何保证exactly-once，这个时候可以说说checkpoint的机制。还有一个是端到端的 exactly-once，这个可以把第一个面试题的答案说说一遍。

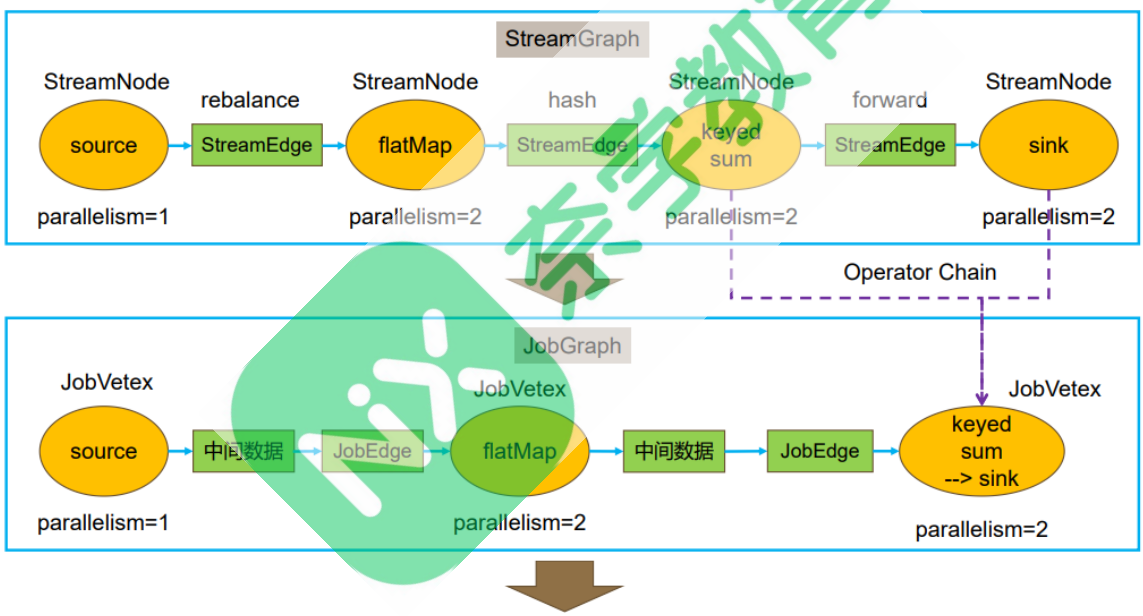
8.Flink 的 checkpoint 机制对比 spark 有什么不同和优势?

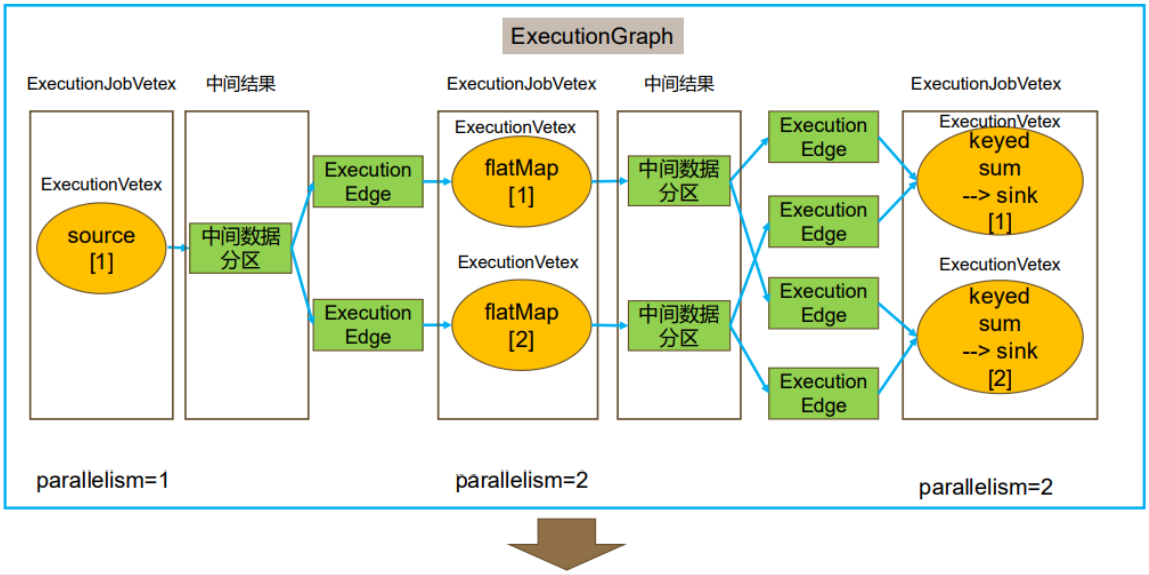
spark streaming 的 checkpoint 仅仅是针对 driver 的故障恢复做了数据和元数据的 checkpoint。而 flink 的 checkpoint 机制要复杂了很多，它采用的是轻量级的分布式快照，实现了每个算子的快照，及流动中的数据的快照。

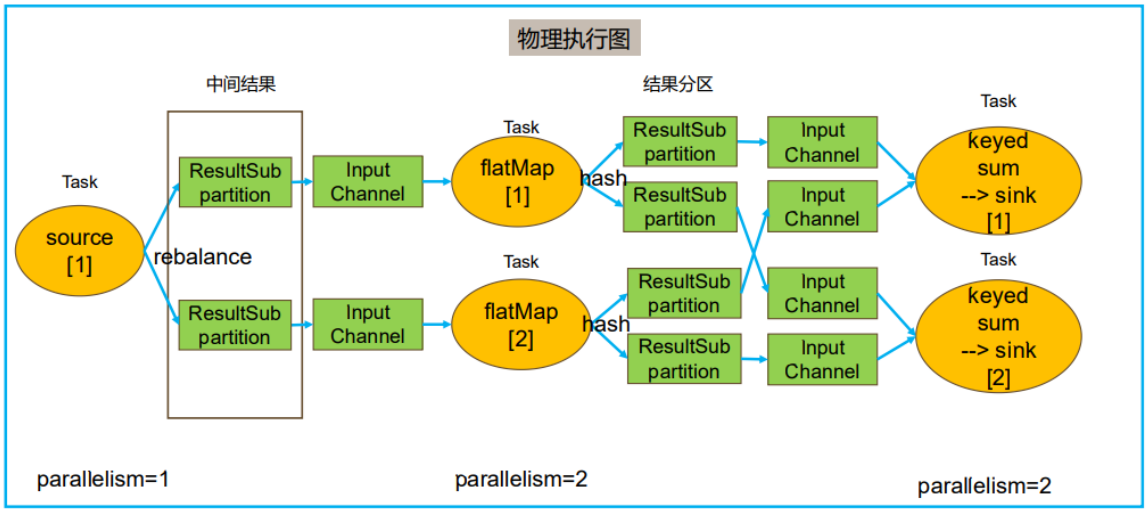
9. Flink怎么去重？考虑一个实时场景：双十一场景，滑动窗口长度为 1 小时， 滑动距离为 10 秒钟，亿级用户，怎样计算 UV?

使用类似于 scala 的 set 数据结构或者 redis 的 set 显然是不行的，因为可能有上亿个 Key，内存放不下。所以可以考虑使用布隆过滤器（Bloom Filter）来去重。

10. 说说flink的三层模型

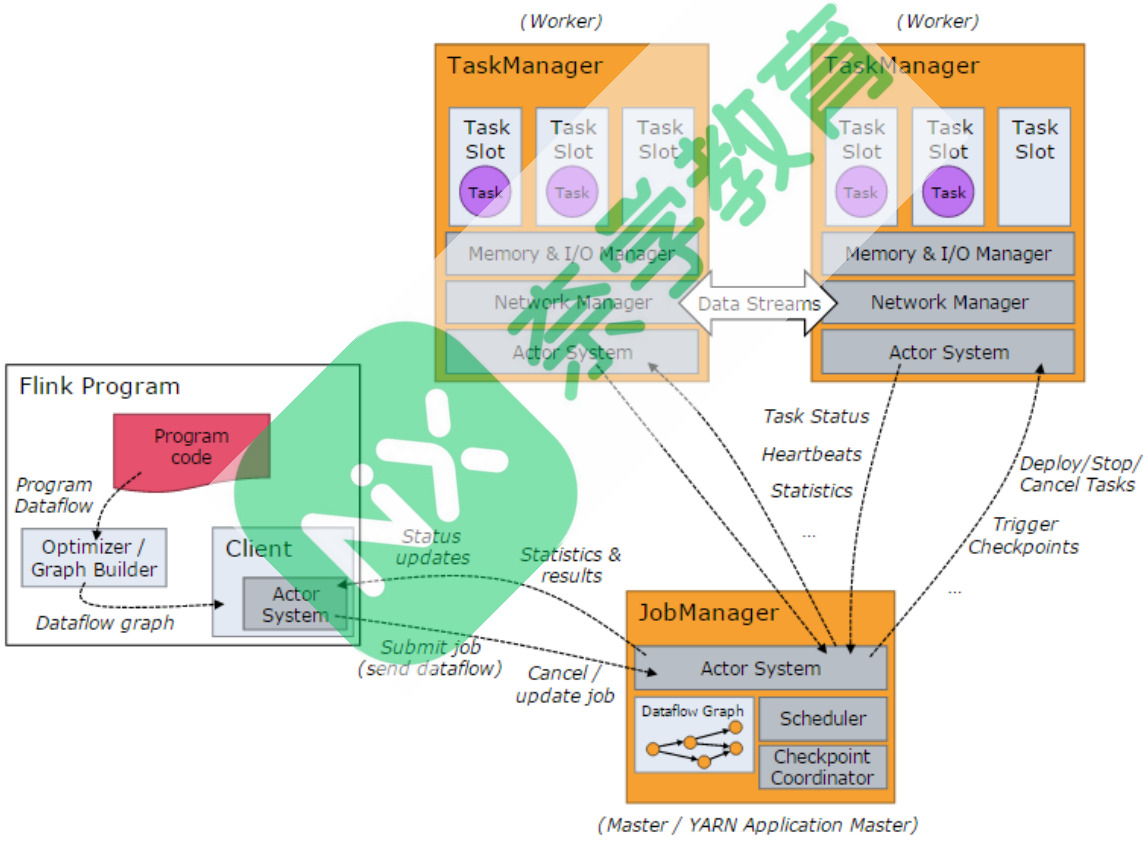






参考建议：他问的是三层模型，其实平时我们讲的是四层模型，都是一样的，你可以把三个图说一下。然后再结合Flink任务的提交流程描述一下即可，千万不要把这个面试想复杂了。

11. 说说Flink任务的运行流程



参考建议：把整个流程描述一下即可

12.如果Flink任务延迟高，你有什么优化的思路?

最主要的手段是资源调优和算子调优。资源调优即是对作业中的Operator的并发数（parallelism）、CPU（core）、堆内存（heap_memory）等参数进行调优。作业参数调优包括：并行度的设置，State的设置，checkpoint的设置

参考建议：从多几个角度分析，回答得有理有据即可，这个是个开放性的题目，没有标准答案。

13.Flink是如何处理反压的

Flink 内部是基于 producer-consumer 模型来进行消息传递的，Flink的反压设计也是基于这个模型。Flink 使用了高效有界的分布式阻塞队列，就像 Java 通用的阻塞队列（BlockingQueue）一样。下游消费者消费变慢，上游就会受到阻塞。

14. Flink里面有哪些分区策略?

GlobalPartitioner
数据会被分发到下游算子的第一个实例中进行处理。

ShufflePartitioner
数据会被随机分发到下游算子的每一个实例中进行处理。

RebalancePartitioner
数据会被循环发送到下游的每一个实例中进行处理。

RescalePartitioner
这种分区器会根据上下游算子的并行度，循环的方式输出到下游算子的每个实例。
这里有点难以理解，假设上游并行度为2，编号为A和B。下游并行度为4，编号为1，2，3，4。
那么A则把数据循环发送给1和2，B则把数据循环发送给3和4。
假设上游并行度为4，编号为A，B，C，D。下游并行度为2，编号为1，2。那么A和B则把数据发送给1，C和D则把数据发送给2。

BroadcastPartitioner
广播分区会将上游数据输出到下游算子的每个实例中。适合于大数据集和小数据集做Jion的场景。

ForwardPartitioner
ForwardPartitioner 用于将记录输出到下游本地的算子实例。它要求上下游算子并行度一样。
简单的说，ForwardPartitioner用来做数据的控制台打印。

KeyGroupStreamPartitioner
Hash分区器。会将数据按 Key 的 Hash 值输出到下游算子实例中。

CustomPartitionerWrapper
用户自定义分区器。需要用户自己实现Partitioner接口，来定义自己的分区逻辑。

15. JobManger在集群中扮演了什么角色

JobManager 负责整个 Flink 集群任务的调度以及资源的管理，从客户端中获取提交的应用，然后根据集群中 TaskManager 上 TaskSlot 的使用情况，为提交的应用分配相应的 TaskSlot 资源并命令 TaskManager 启动从客户端中获取的应用。
JobManager 相当于整个集群的 Master 节点，且整个集群有且只有一个活跃的 JobManager ，负责整个集群的任务管理和资源管理。
JobManager 和 TaskManager 之间通过 Actor System 进行通信，获取任务执行的情况并通过 Actor System 将应用的任务执行情况发送给客户端。
同时在任务执行的过程中，Flink JobManager 会触发 Checkpoint 操作，每个 TaskManager 节点收到 Checkpoint 触发指令后，完成 Checkpoint 操作，所有的 Checkpoint 协调过程都是在 Fink JobManager 中完成。
当任务完成后，Flink 会将任务执行的信息反馈给客户端，并且释放掉 TaskManager 中的资源以供下一次提交任务使用。

16. Operator Chains（算子链）这个概念你了解吗

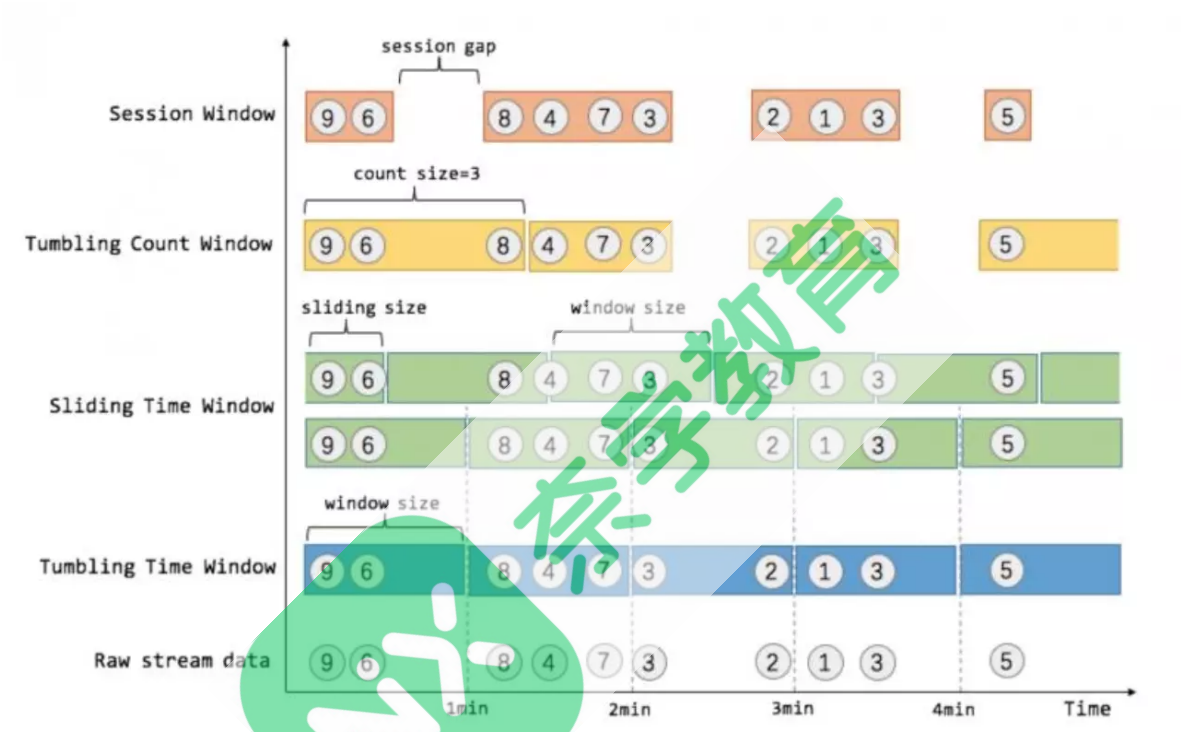
为了更高效地分布式执行，Flink会尽可能地将operator的subtask链接（chain）在一起形成task。每个task在一个线程中执行。将operators链接成task是非常有效的优化：它能减少线程之间的切换，减少消息的序列化/反序列化，减少数据在缓冲区的交换，减少了延迟的同时提高整体的吞吐量。这就是我们所说的算子链。

17. Flink 中水印是什么概念，起到什么作用?

Watermark 是 Apache Flink 为了处理 EventTime 窗口计算提出的一种机制，本质上是一种时间戳。一般来讲Watermark经常和Window一起被用来处理乱序事件。

参考建议：根据自己的理解说明即可，如果结合实际的例子说明更佳。

18. 说说Flink的窗口



Flink 支持两种划分窗口的方式，按照time和count。如果根据时间划分窗口，那么它就是一个time-window 如果根据数据划分窗口，那么它就是一个count-window。

flink支持窗口的两个重要属性（size和interval）

如果size=interval,那么就会形成tumbling-window(无重叠数据)
如果size>interval,那么就会形成sliding-window(有重叠数据)
如果size< interval, 那么这种窗口将会丢失数据。比如每5秒钟，统计过去3秒的通过路口汽车的数据，将会漏掉2秒钟的数据。

通过组合可以得出四种基本窗口：

time-tumbling-window 无重叠数据的时间窗口，设置方式举例：timeWindow(Time.seconds(5))
time-sliding-window 有重叠数据的时间窗口，设置方式举例：timeWindow(Time.seconds(5), Time.seconds(3))
count-tumbling-window无重叠数据的数量窗口，设置方式举例：countWindow(5)
count-sliding-window 有重叠数据的数量窗口，设置方式举例：countWindow(5,3)

参考建议：这个题回答起来难度不大，但是回答的时候一定要注意思路要清晰，按照类别说明，这个题难度不大，但是很容易自己把自己都说乱了。